



IP0390.I

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appeal No. \_\_\_\_\_

Application No.: 10/646,289  
Filing Date: August 21, 2003  
Appellants: Son Ho et al.  
Conf. No.:  
Group Art Unit: 2188  
Examiner: Kaushikkumar Patel  
Title: LINE CACHE CONTROLLER WITH LOOKAHEAD

---

**BRIEF ON APPEAL ON BEHALF OF APPELLANTS AND PETITION FOR  
EXTENSION OF TIME**

Mail Stop Appeal Brief-Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

March 6, 2008

Sir:

This appeal is from the decision of the Patent Examiner dated May 7, 2007, rejecting claims 1-2, 4-7, 9-10, 12-15, and 17-33, which are reproduced in Appendix A of this Appeal Brief.

Appellants hereby petition under the provisions of 37 C.F.R. § 1.136(a) for an extension of time in which to respond to the outstanding Office Action and includes a fee as set forth in 37 C.F.R. § 1.17(a) with this response for such extension of time.

03/10/2008 RFEKADU1 00000028 10646289

02 FC:1402

510.00 DP

## TABLE OF CONTENTS

Application No.: 10/646,289.....	1
I. REAL PARTY IN INTEREST.....	3
II. RELATED APPEALS AND INTERFERENCES.....	3
III. STATUS OF THE CLAIMS.....	3
IV. STATUS OF THE AMENDMENTS.....	3
V. SUMMARY OF THE CLAIMED SUBJECT MATTER.....	4
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL.....	8
VII. ARGUMENTS.....	9
A. The Rejections.....	9
B. Claim Distinctions.....	9
1. Distinctions regarding independent Claims 1, 5, 10, 13, and 18.....	9
2. Dependent Claims 2, 4, 6-7, 9, 12, 14-15, 17, and 19-25.....	13
3. Distinctions regarding independent Claim 26.....	13
4. Dependent Claims 27-33.....	17
VIII. CONCLUSION.....	17
IX. APPENDIX A.....	19
CLAIMS APPENDED.....	19
X. APPENDIX B.....	29
EVIDENCE APPENDED.....	29
XI. APPENDIX C.....	29
RELATED PROCEEDINGS APPENDED.....	29

**BRIEF ON APPEAL ON BEHALF OF APPELLANTS**

In support of the Notice of Appeal filed September 5, 2007, appealing the Examiner's Rejection of each of claims 1-2, 4-7, 9-10, 12-15, and 17-33, mailed May 7, 2007, which appear in the attached Appendix A, Appellants hereby provide the following remarks.

**I. REAL PARTY IN INTEREST**

The present application is assigned to Marvell International, Ltd as recorded in the Patent and Trademark Office at Reel 014449, Frame 0027 and Reel 014448, Frame 0988.

**II. RELATED APPEALS AND INTERFERENCES**

An appeal is pending in related Patent Application No. 10/626,507. The undersigned, the Assignee, and the Appellants do not know of any other appeals or interferences which would directly affect or that would be directly affected by, or have a bearing on, the Board's decision in this Appeal.

**III. STATUS OF THE CLAIMS**

Claims 1-2, 4-7, 9-10, 12-15, and 17-33 are reproduced in the attached Appendix A and are the claims on Appeal. Each of these claims stands rejected.

**IV. STATUS OF THE AMENDMENTS**

There are no pending amendments filed subsequent to the final rejection.

## **V. SUMMARY OF THE CLAIMED SUBJECT MATTER**

Independent claim 1 recites a memory storage system (see FIG. 1) that is accessed by a first central processing unit (CPU) (see FIG. 1, element 50, Page 6, Lines 20-21). The memory storage system includes a line cache (see FIG. 1, element 58; Page 7, Line 1) including a plurality of pages that are accessed by the first CPU. A first memory device (see FIG. 25, element 550; Page 26, Lines 13-15) stores data that is loaded into said line cache when a miss occurs. When said miss occurs and before a second miss occurs,  $n$  pages of said line cache are loaded with data from sequential locations in said first memory device, wherein  $n$  is greater than one (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9). A line cache control system (see FIG. 1) controls data flow between said line cache, the first CPU, said first memory device and a second memory device (see FIG. 25, element 540; Page 26, Lines 13-15).

The line cache control system includes a first line cache interface (see FIG. 1, element 52; Page 6, Line 21) that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request, a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) that communicates with said first memory device, a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11) that communicates with said second memory device, and a switch (see FIG. 1, element 64; Page 7, Lines 1-3) that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said  $n$  pages of said line cache when said miss occurs.

Independent claim 5 recites a memory storage system (see FIG. 1) that is accessed by a first central processing unit (CPU) (see FIG. 1, element 50, Page 6, Lines 20-21). The memory storage system includes a line cache (see FIG. 1, element 58; Page 7, Line 1) including a plurality of pages that are accessed by the first CPU. A first memory device (see FIG. 25, element 550; Page 26, Lines 13-15) stores data that

is loaded into said line cache when a miss occurs. After an initial miss, said line cache prevents any additional misses as long as the first CPU addresses sequential memory locations of said first memory device (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9). A line cache control system (see FIG. 1) controls data flow between said line cache, the first CPU, said first memory device and a second memory device (see FIG. 25, element 540; Page 26, Lines 13-15).

The line cache control system includes a first line cache interface (see FIG. 1, element 52; Page 6, Line 21) that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request, a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) that communicates with said first memory device, a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11) that communicates with said second memory device, and a switch (see FIG. 1, element 64; Page 7, Lines 1-3) that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

Independent claim 10 recites a memory storage system (see FIG. 1) that includes a line cache (see FIG. 1, element 58; Page 7, Line 1) including a plurality of pages. A first central processing unit (CPU) (see FIG. 1, element 50, Page 6, Lines 20-21) accesses data stored in said line cache. A first memory device (see FIG. 25, element 550; Page 26, Lines 13-15) stores data that is loaded into said line cache when a miss occurs. When said miss occurs and before a second miss occurs, n pages of said line cache are loaded with data from sequential locations in said first memory device, wherein n is greater than one (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9), and when said first CPU requests data from an m<sup>th</sup> page of said n pages in said line cache, wherein m is greater than one and less than or equal to n, said line cache loads p additional pages with data from sequential locations in said first memory device (see FIG. 24, element 520; Page 25, Lines 20-23). A line cache control system (see FIG. 1) controls data flow between said line cache, the first CPU, said first

memory device and a second memory device (see FIG. 25, element 540; Page 26, Lines 13-15).

The line cache control system includes a first line cache interface (see FIG. 1, element 52; Page 6, Line 21) that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request, a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) that communicates with said first memory device, a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11) that communicates with said second memory device, and a switch (see FIG. 1, element 64; Page 7, Lines 1-3) that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

Independent claim 13 recites a memory storage system (see FIG. 1) that includes a line cache (see FIG. 1, element 58; Page 7, Line 1) including a plurality of pages. A first central processing unit (CPU) (see FIG. 1, element 50, Page 6, Lines 20-21) accesses data stored in said line cache. A first memory device (see FIG. 25, element 550; Page 26, Lines 13-15) stores data that is loaded into said line cache when a miss occurs. After an initial miss, said line cache prevents any additional misses as long as said first CPU addresses sequential memory locations of said first memory device (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9). A line cache control system (see FIG. 1) controls data flow between said line cache, the first CPU, said first memory device and a second memory device (see FIG. 25, element 540; Page 26, Lines 13-15).

The line cache control system includes a first line cache interface (see FIG. 1, element 52; Page 6, Line 21) that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request, a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) that communicates with said first memory device, a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11) that communicates with said second

memory device, and a switch (see FIG. 1, element 64; Page 7, Lines 1-3) that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

Independent claim 18 recites a storage system that includes a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) and a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11). A first memory (see FIG. 1, element 66) communicates exclusively with said first memory interface. A second memory (see FIG. 1, element 70) communicates exclusively with said second memory interface. A cache (see FIG. 1, element 58; Page 7, Line 1) stores data from said first and second memories and receives a data request from a processor (see FIG. 1, element 50, Page 6, Lines 20-21) specifying a first address, wherein when requested data corresponding to said first address is present in said cache, said cache provides said requested data to the processor.

A switch (see FIG. 1, element 64; Page 7, Lines 1-3) communicates with said cache, wherein when said requested data is not present in said cache, said switch exclusively connects one of said first memory interface and said second memory interface to said cache, as selected by said first address, whereby said cache retrieves said requested data. After a first time that said requested data is not present in said cache, n pages of said cache are loaded with data from sequential locations of one of said first and second memory devices to prevent any additional cache misses for as long as sequential memory locations of said one of said first and second memory devices are addressed (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9).

Independent claim 26 recites a storage system that includes a first memory interface (see FIG. 1, element 78; Page 7, Lines 4-5) and a second memory interface (see FIG. 1, element 80; Page 7, Lines 10-11). A first memory (see FIG. 1, element 66) communicates exclusively with said first memory interface. A second memory (see FIG. 1, element 70) communicates exclusively with said second memory interface. An arbitration module (see FIG. 2, element 100; Page 8, Lines 9-11) receives data requests

from first (see FIG. 1, element 50, Page 6, Lines 20-21) and second (see FIG. 2, element 50-2; Page 8, Lines 4-6) processors, and processes said data requests into ordered data requests. A cache (see FIG. 1, element 58; Page 7, Line 1) stores data from said first and second memories and receives one of said ordered data requests specifying a first address and associated with one processor of the first and second processors, wherein when requested data corresponding to said first address is present in said cache, said cache provides said requested data to the one processor.

A switch (see FIG. 1, element 64; Page 7, Lines 1-3) communicates with said cache, wherein when said requested data is not present in said cache, said switch exclusively connects one of said first memory interface and said second memory interface to said cache, as selected by said first address, whereby said cache retrieves said requested data. After a first time that said requested data is not present in said cache, n pages of said cache are loaded with data from sequential locations of one of said first and second memory devices to prevent any additional cache misses for as long as sequential memory locations of said one of said first and second memory devices are addressed (see FIG. 24, element 508; Page 25, Line 15-Page 26, Line 9).

## **VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL**

Appellants seek the Board's review of the rejection of:

(a) Claims 1-2, 5-7, 10, 13-15, 18-19 and 25 under 35 U.S.C. § 103(a) as being unpatentable over Zaidi et al. (U.S. Pat. No. 6,601,126 B1), Jeddelloh (U.S. Pat. No. 7,133,972 B2) and Loafman (U.S. Pub. No. 2005/0021916 A1); and

(b) Claims 4, 9, 12, 17, 26 and 33 under 35 U.S.C. § 103(a) as being unpatentable over Zaidi et al. (U.S. Pat. No. 6,601,126 B1), Jeddelloh (U.S. Pat. No. 7,133,972 B2) and Loafman (U.S. Pub. No. 2005/0021916 A1) as applied to claims 1-2, 5-7 and 13-15 above, and further in view of Barroso et al. (U.S. Pat. No. 6,725,334 B2).



## **VII. ARGUMENTS**

### **A. The Rejections**

The rejections that are the subject of this appeal are: a rejection of each of independent claims 1, 5, 10, 13, 18, and 26 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Zaidi in view of one or more of Jeddelloh, Loafman, and Barroso.

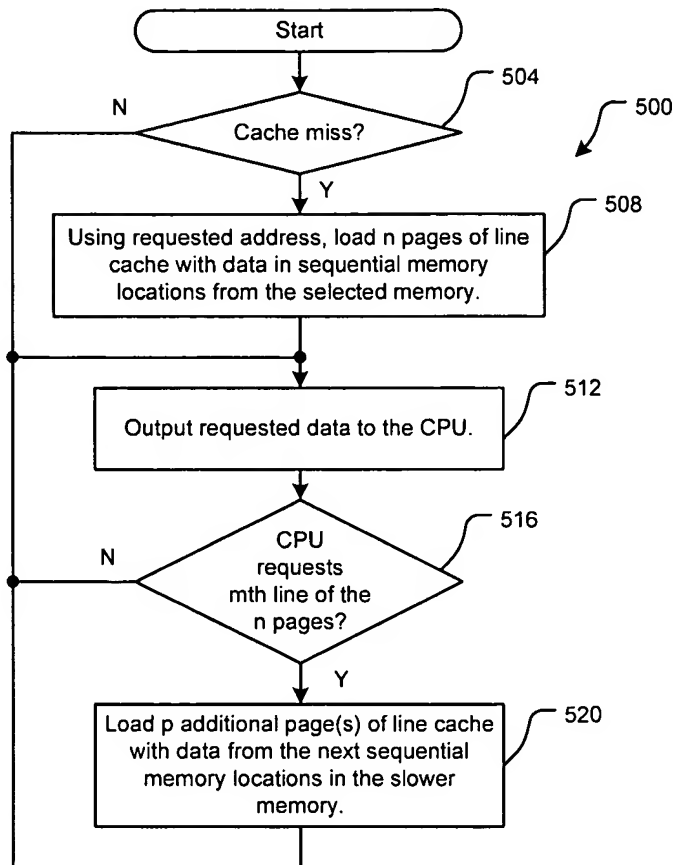
With respect to independent claims 1, 5, 10, 13, 18, and 26, the Examiner admits that Zaidi fails to teach loading *n* pages from sequential locations from memory, and instead relies on Loafman to disclose this limitation, citing Paragraphs [0026] and [0011]-[0013] of Loafman (see Page 5, Line 21 through Page 6, Line 8 of the Office Action mailed May 7, 2007). The Examiner **admits that neither Zaidi nor Loafman discloses loading *n* pages of a line cache after a first cache miss and before a second miss.** Instead, the Examiner notes that Loafman teaches that “applications/programs read data either sequentially or randomly...and if data [is] being read randomly, prefetching may not work,” citing Paragraph [0013] of Loafman (see Page 6, Line 9 through Page 8, Line 6 of the Office Action mailed May 7, 2007).

### **B. Claim Distinctions**

#### **1. Distinctions regarding independent Claims 1, 5, 10, 13, and 18**

With respect to claim 1, Appellants respectfully submit that Zaidi, either singly or in combination with Jeddelloh and Loafman, fail to at least show, teach, or suggest a line cache including a plurality of pages that are accessed by the first CPU and a first memory device that stores data that is loaded into said line cache when a miss occurs, wherein when said miss occurs **and before a second miss occurs**, *n* pages of said line cache are loaded with data from sequential locations in said first memory device, wherein *n* is greater than one. Loafman appears to disclose that at least two misses occur before prefetching data from multiple sequential memory locations.

The present claims are directed to loading the cache with data from sequential memory locations after a first cache miss and before a second cache miss. As described in an exemplary embodiment in FIG. 24 of the present application, a **cache miss** occurs in step 504:



**FIG. 24**

When the cache miss occurs, n (wherein n is greater than one) pages of line cache are loaded with data in sequential memory locations from a selected memory.

Appellants respectfully note that the sequential memory locations are loaded after this initial miss and before any other misses occur. For example, “the look ahead method 500 according to the present invention ensures that as long as the CPU continues to access sequential memory locations, **there will never be a miss after the initial miss.**” (See Paragraph [0093] of the present application; Emphasis added). In other words, the n pages of the line cache are loaded with the data from the sequential

locations in response to an initial cache miss and before a second (i.e. any additional) cache miss.

The Examiner acknowledges that Zaidi fails to disclose this limitation. Instead, the Examiner alleges that Loafman discloses preloading consecutive pages from lower latency storage into memory when a miss occurs in memory at Paragraph [0026]. The cited portion of Loafman states that a Virtual Memory Manager (VMM) 112 “anticipates future needs for pages of data of a file **by observing the pattern used by a program** that is accessing the file.” (Emphasis added). More specifically, “[w]hen the program accesses two successive pages... each using a page fault, the VMM 112 assumes that the program will continue to access the data sequentially.” Appellants respectfully note that a page fault occurs in response to a miss (see Paragraph [0023]). In other words, Loafman discloses retrieving the consecutive pages after observing a pattern of successive page accesses. As such, **at least two misses are required**.

Further, during a telephonic interview on March 5, 2007, the Examiner noted that Loafman discloses that pre-fetching is known in the art. For example, Paragraph [0012] recites that “pre-fetching works splendidly when data is being read sequentially” and notes that “after two consecutive page faults of sequentially stored data, a block of sequential pages of data will be pre-fetched.” In other words, Appellants respectfully note that Loafman still discloses that the pre-fetching is only performed after two consecutive misses (i.e. page faults).

Similarly, Paragraph [0013] states:

However, if data is being read randomly, spatial data pre-fetching may not work as well. For example, **suppose an executing program is randomly reading data. Suppose further that the executing program makes a request to read a certain amount of data that resides on two sequential pages. If the data is not already in RAM, two page faults will be raised in order to load the two pages in the RAM. Because the pages are sequential, the system may infer that data is being read sequentially; and hence, pre-fetch a block of sequential pages of data.** Since data is being read randomly, it is highly unlikely that future needed data will be on the pre-fetched block of pages. Thus, the block of pages may have been pre-fetched in vain and the physical pages onto which they are placed wasted. As will be explained later, continually pre-fetching unneeded pages of data may place an undue pressure on RAM space.

Here, Loafman discloses that when random reading of data coincidentally results in **misses of two consecutive pages**, pre-fetching may be performed. In other words, **regardless of whether data is being read randomly or sequentially, Loafman still appears to disclose that two misses are required for pre-fetching.** Loafman appears to be absent of any teaching or suggestion of loading data from sequential locations before a second cache miss.

Here again, Appellants respectfully submit that the cited portion discloses retrieving the consecutive pages after observing **a pattern of successive page accesses**. As such, **at least two misses are required**, which is not analogous to “before a second miss occurs” as claim 1 recites.

The Examiner appears to acknowledge that Loafman discloses prefetching sequentially after at least two page faults (i.e. misses). For example,

It is apparently clear from the above statements that Loafman is not prefetching pages if data [is] being read randomly, but since [the] program requests two sequential pages (even though program accesses data randomly, it is not totally random and can access pages sequentially), two page faults will be raised. (See Page 6, Lines 16-22 of the Office Action).

In other words, the Examiner’s interpretation appears to be analogous to Appellants’ position that Loafman discloses that **at least two cache misses are required** before any sequential prefetching. The Examiner further notes (at Page 7, Lines 15-16 of the Office Action):

Thus, it is apparently clear that Loafman uses two page faults (two cache misses) to two sequential pages to confirm that data is being read sequentially.

As such, Appellants respectfully submit that the combination of Zaidi and Loafman appear to be absent of any teaching or suggestion of loading multiple pages of the line cache from sequential memory locations in response to a cache miss and before a second cache miss. Instead, the Examiner alleges that it would be obvious to modify Loafman in this manner “if the program reads data sequentially...since it is known that data is being read sequentially.”

Appellants respectfully note that none of the references, in particular Loafman, teach or suggest any “prior knowledge” of sequential data reading that the Examiner

appears to suggest. In contrast, Loafman clearly discloses that at least two cache misses are required to determine whether a program reads data sequentially. In other words, according to Loafman, a determination that a program is reading data sequentially requires at least two consecutive cache misses. **The Examiner fails to provide any reference that teaches or suggests any knowledge of sequential data reading that is not based on at least two cache misses.**

Appellants respectfully submit that the combination of Zaidi and Loafman appear to be absent of any teaching or suggestion of loading multiple pages of the line cache from sequential memory locations in response to a cache miss and before a second cache miss. Accordingly, Appellants respectfully submit that claim 1, as well as its dependent claims, should be in condition for allowance for at least the above reasons. Claims 10, 5, 13, and 18, as well as their corresponding dependent claims, should be in condition for allowance for at least similar reasons.

## **2. Dependent Claims 2, 4, 6-7, 9, 12, 14-15, 17, and 19-25**

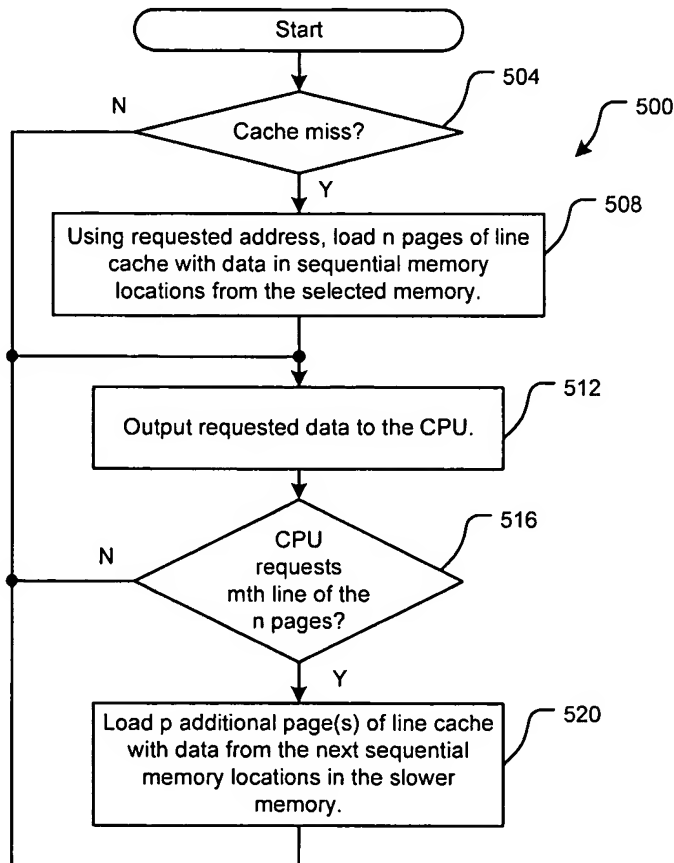
With regard to claims 2, 4, 6-7, 9, 12, 14-15, 17, and 19-25, these claims are allowable for at least the reasons previously presented with regard to claims 1, 5, 10, 13, and 18, respectively. Accordingly, it is respectfully requested that the rejection of these claims be overturned.

## **3. Distinctions regarding independent Claim 26**

With respect to claim 1, Appellants respectfully submit that Zaidi, either singly or in combination with Jeddeloh and Loafman, fail to at least show, teach, or suggest that **after a first time** that said requested data is not present in said cache, n pages of said cache are loaded with data from sequential locations of one of said first and second memory devices **to prevent any additional cache misses** for as long as sequential memory locations of said one of said first and second memory devices are addressed.

Instead, Loafman appears to disclose that at least two misses occur before prefetching data from multiple sequential memory locations.

The present claims are directed to loading the cache with data from sequential memory locations after a first cache miss and before a second cache miss. As described in an exemplary embodiment in FIG. 24 of the present application, a **cache miss** occurs in step 504:



**FIG. 24**

When the cache miss occurs, n (wherein n is greater than one) pages of line cache are loaded with data in sequential memory locations from a selected memory.

Appellants respectfully note that the sequential memory locations are loaded after this initial miss and before any other misses occur. For example, “the look ahead method 500 according to the present invention ensures that as long as the CPU continues to access sequential memory locations, **there will never be a miss after the**

**initial miss.”** (See Paragraph [0093] of the present application; Emphasis added). In other words, the *n* pages of the line cache are loaded with the data from the sequential locations **after a first miss and before any additional cache misses.**

The Examiner acknowledges that Zaidi fails to disclose this limitation. Instead, the Examiner alleges that Loafman discloses preloading consecutive pages from lower latency storage into memory when a miss occurs in memory at Paragraph [0026]. The cited portion of Loafman states that a Virtual Memory Manager (VMM) 112 “anticipates future needs for pages of data of a file **by observing the pattern used by a program** that is accessing the file.” (Emphasis added). More specifically, “[w]hen the program accesses two successive pages... each using a page fault, the VMM 112 assumes that the program will continue to access the data sequentially.” Appellants respectfully note that a page fault occurs in response to a miss (see Paragraph [0023]). In other words, Loafman discloses retrieving the consecutive pages after observing a pattern of successive page accesses. As such, **at least two misses are required.**

Further, during a telephonic interview on March 5, 2007, the Examiner noted that Loafman discloses that pre-fetching is known in the art. For example, Paragraph [0012] recites that “pre-fetching works splendidly when data is being read sequentially” and notes that “after two consecutive page faults of sequentially stored data, a block of sequential pages of data will be pre-fetched.” In other words, Appellants respectfully note that Loafman still discloses that the pre-fetching is only performed after two consecutive misses (i.e. page faults).

Similarly, Paragraph [0013] states:

However, if data is being read randomly, spatial data pre-fetching may not work as well. For example, **suppose an executing program is randomly reading data. Suppose further that the executing program makes a request to read a certain amount of data that resides on two sequential pages. If the data is not already in RAM, two page faults will be raised in order to load the two pages in the RAM. Because the pages are sequential, the system may infer that data is being read sequentially; and hence, pre-fetch a block of sequential pages of data.** Since data is being read randomly, it is highly unlikely that future needed data will be on the pre-fetched block of pages. Thus, the block of pages may have been pre-fetched in vain and the physical pages onto which they are placed wasted. As will be explained later, continually pre-

fetching unneeded pages of data may place an undue pressure on RAM space.

Here, Loafman discloses that when random reading of data coincidentally results in **misses of two consecutive pages**, pre-fetching may be performed. In other words, **regardless of whether data is being read randomly or sequentially, Loafman still appears to disclose that two misses are required for pre-fetching.** Loafman appears to be absent of any teaching or suggestion of loading data from sequential locations before a second cache miss.

Here again, Appellants respectfully submit that the cited portion discloses retrieving the consecutive pages after observing **a pattern of successive page accesses**. As such, **at least two misses are required**, which is not analogous to “after a first time...to prevent any additional cache misses” as claim 26 recites.

The Examiner appears to acknowledge that Loafman discloses prefetching sequentially after at least two page faults (i.e. misses). For example,

It is apparently clear from the above statements that Loafman is not prefetching pages if data [is] being read randomly, but since [the] program requests two sequential pages (even though program accesses data randomly, it is not totally random and can access pages sequentially), two page faults will be raised. (See Page 6, Lines 16-22 of the Office Action).

In other words, the Examiner’s interpretation appears to be analogous to Appellants’ position that Loafman discloses that **at least two cache misses are required** before any sequential prefetching. The Examiner further notes (at Page 7, Lines 15-16 of the Office Action):

Thus, it is apparently clear that Loafman uses two page faults (two cache misses) to two sequential pages to confirm that data is being read sequentially.

As such, Appellants respectfully submit that the combination of Zaidi and Loafman appear to be absent of any teaching or suggestion of loading multiple pages of the line cache from sequential memory locations in response to a cache miss and before a second cache miss. Instead, the Examiner alleges that it would be obvious to modify Loafman in this manner “if the program reads data sequentially...since it is known that data is being read sequentially.”



Appellants respectfully note that none of the references, in particular Loafman, teach or suggest any “prior knowledge” of sequential data reading that the Examiner appears to suggest. In contrast, Loafman clearly discloses that at least two cache misses are required to determine whether a program reads data sequentially. In other words, according to Loafman, a determination that a program is reading data sequentially requires at least two consecutive cache misses. **The Examiner fails to provide any reference that teaches or suggests any knowledge of sequential data reading that is not based on at least two cache misses.**

Appellants respectfully submit that the combination of Zaidi and Loafman appear to be absent of any teaching or suggestion of loading multiple pages of the line cache from sequential memory locations in response to a cache miss and before a second cache miss. Accordingly, Appellants respectfully submit that claim 26, as well as its dependent claims, should be in condition for allowance for at least the above reasons.

#### **4. Dependent Claims 27-33**

With regard to claims 27-33 these claims are allowable for at least the reasons previously presented with regard to claim 26. Accordingly, it is respectfully requested that the rejection of these claims be overturned.

#### **VIII. CONCLUSION**

Appellants respectfully request the Honorable Board of Patent Appeals and Interferences to reverse the Examiner’s rejection of each of pending claims 11-2, 4-7, 9-10, 12-15, and 17-33. Appellants respectfully submit that the prior art does not teach or suggest one or more limitations of the claims as discussed above. Accordingly, for at least the aforementioned reasons, Appellants respectfully request the Honorable members of the Board of Patent Appeals and Interferences to reverse the outstanding rejections in connection with the present application and permit each of claims 1-2, 4-7, 9-10, 12-15, and 17-33 to be passed to allowance in connection with the present application.

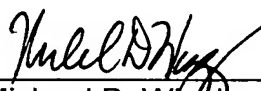
Should there be any outstanding matters that need to be resolved in the present application, the Examiner is respectfully requested to contact Michael D. Wiggins, Reg. No. 34,754, or Damian M. Aquino, Reg. No. 54,964, at the telephone number of the undersigned below.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No. 08-0750 for any additional fees required under 37 C.F.R. § 1.16 or under 37 C.F.R. § 1.17; particularly, extension of time fees.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By:

  
\_\_\_\_\_  
Michael D. Wiggins  
Reg. No. 34,754  
Damian M. Aquino  
Reg. No. 54,964

MDW/DMA/dms

**Please address all correspondence to:**

**Harness, Dickey & Pierce, P.L.C.**

**5445 Corporate Drive**

**Suite 200**

**Troy, MI 48098**

**Customer No. 26703**

**Tel. No. (248) 641-1600**

**Fax. No. (248) 641-0270**

**IX. APPENDIX A****CLAIMS APPENDED**

This is a complete and current listing of the claims.

1. (Previously Presented) A memory storage system that is accessed by a first central processing unit (CPU), comprising:
  - a line cache including a plurality of pages that are accessed by the first CPU;
  - a first memory device that stores data that is loaded into said line cache when a miss occurs,
    - wherein when said miss occurs and before a second miss occurs, n pages of said line cache are loaded with data from sequential locations in said first memory device, wherein n is greater than one;
  - a second memory device; and
  - a line cache control system that controls data flow between said line cache, the first CPU, said first memory device and said second memory device, and that includes:
    - a first line cache interface that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request;
    - a first memory interface that communicates with said first memory device;
    - a second memory interface that communicates with said second memory device; and

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

2. (Original) The memory storage system of claim 1 wherein when the first CPU requests data from an  $m^{\text{th}}$  page of said n pages in said line cache, wherein m is greater than one and less than or equal to n, said line cache loads p additional pages with data from sequential locations in said first memory device.

3. (Cancelled).

4. (Previously Presented) The memory storage system of claim 1 further comprising:

a second CPU;

a second line cache interface that is associated with said second CPU, that receives a second program read request from said second CPU and that generates a second address from said second program read request; and

a line cache arbitration device that communicates with said first and second line cache interfaces and said line cache and that resolves line cache access conflicts between the first CPU and said second CPU.

5. (Previously Presented) A memory storage system that is accessed by a first central processing unit (CPU), comprising:

a line cache including a plurality of pages that are accessed by the first CPU;

a first memory device that stores data that is loaded into said line cache when a miss occurs,

wherein after an initial miss, said line cache prevents any additional misses as long as the first CPU addresses sequential memory locations of said first memory device;

a second memory device; and

a line cache control system that controls data flow between said line cache, the first CPU, said first memory device and said second memory device, and that includes:

a first line cache interface that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request;

a first memory interface that communicates with said first memory device;

a second memory interface that communicates with said second memory device; and

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

6. (Original) The memory storage system of claim 5 wherein when said miss occurs, n pages of said line cache are loaded with data from sequential locations in said first memory device, wherein n is greater than one.

7. (Original) The memory storage system of claim 6 wherein when the first CPU requests data from an  $m^{\text{th}}$  page of said n pages in said line cache, wherein m is greater than one and less than or equal to n, said line cache loads p additional pages with data from sequential locations in said first memory device.

8. (Cancelled).

9. (Previously Presented) The memory storage system of claim 5 further comprising:

a second CPU;

a second line cache interface that is associated with said second CPU, that receives a second program read request from said second CPU and that generates a second address from said second program read request; and

a line cache arbitration device that communicates with said first and second line cache interfaces and said line cache and that resolves line cache access conflicts between the first CPU and said second CPU.

10. (Previously Presented) A memory storage system, comprising:

a line cache including a plurality of pages;

a first central processing unit (CPU) that accesses data stored in said line cache;

a first memory device that stores data that is loaded into said line cache when a miss occurs,

wherein when said miss occurs and before a second miss occurs,  $n$  pages of said line cache are loaded with data from sequential locations in said first memory device, wherein  $n$  is greater than one, and wherein when said first CPU requests data from an  $m^{\text{th}}$  page of said  $n$  pages in said line cache, wherein  $m$  is greater than one and less than or equal to  $n$ , said line cache loads  $p$  additional pages with data from sequential locations in said first memory device;

a second memory device; and

a line cache control system that controls data flow between said line cache, said first CPU, said first memory device and said second memory device, and that includes:

a first line cache interface that is associated with said first CPU, that receives a first program read request from said first CPU and that generates a first address from said first program read request;

a first memory interface that communicates with said first memory device;

a second memory interface that communicates with said second memory device; and

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and retrieves said n pages of line cache when said miss occurs.

11. (Cancelled).

12. (Previously Presented) The memory storage system of claim 10 further comprising:

a second CPU;

a second line cache interface that is associated with said second CPU, that receives a second program read request from said second CPU and that generates a second address from said second program read request; and

a line cache arbitration device that communicates with said first and second line cache interfaces and said line cache and that resolves line cache access conflicts between said first CPU and said second CPU.

13. (Previously Presented) A memory storage system, comprising:

a line cache including a plurality of pages;

a first central processing unit (CPU) that accesses said pages of said line cache;

a first memory device that stores data that is loaded into said line cache when a miss occurs,

wherein after an initial miss, said line cache prevents any additional misses as long as said first CPU addresses sequential memory locations of said first memory device;

a second memory device; and

a line cache control system that controls data flow between said line cache, said first CPU, said first memory device and said second memory device, and that includes:

a first line cache interface that is associated with said first CPU, that receives a first program read request from said first CPU and that generates a first address from said first program read request;

a first memory interface that communicates with said first memory device;

a second memory interface that communicates with said second memory device; and

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if a match occurs, and loads said n pages of said line cache when said miss occurs.

14. (Original) The memory storage system of claim 13 wherein when said miss occurs, n pages of said line cache are loaded with data from sequential locations in said first memory device, wherein n is greater than one.

15. (Original) The memory storage system of claim 14 wherein when said first CPU requests data from an  $m^{\text{th}}$  page of said n pages in said line cache, wherein m is greater than one and less than or equal to n, said line cache loads p additional pages with data from sequential locations in said first memory device.

16. (Cancelled).

17. (Previously Presented) The memory storage system of claim 13 further comprising:

a second CPU;



a second line cache interface that is associated with said second CPU, that receives a second program read request from said second CPU and that generates a second address from said second program read request; and

a line cache arbitration device that communicates with said first and second line cache interfaces and said line cache and that resolves line cache access conflicts between said first CPU and said second CPU.

18. (Previously Presented) A storage system comprising:

a first memory interface;

a second memory interface;

a first memory that communicates exclusively with said first memory interface;

a second memory that communicates exclusively with said second memory interface;

a cache that stores data from said first and second memories and receives a data request from a processor specifying a first address, wherein when requested data corresponding to said first address is present in said cache, said cache provides said requested data to the processor; and

a switch that communicates with said cache, wherein when said requested data is not present in said cache, said switch exclusively connects one of said first memory interface and said second memory interface to said cache, as selected by said first address, whereby said cache retrieves said requested data,

wherein after a first time that said requested data is not present in said cache, n pages of said cache are loaded with data from sequential locations of one of said first and second memory devices to prevent any additional cache misses for as long as sequential memory locations of said one of said first and second memory devices are addressed.

19. (Previously Presented) The storage system of claim 18 wherein said cache translates an address contained within said data request to obtain said first address.

20. (Previously Presented) The storage system of claim 18 wherein said first memory interface also communicates with the processor via a first direct interface, which bypasses the cache for selected accesses to said first memory.

21. (Previously Presented) The storage system of claim 20 wherein when said first address is located within said first memory, writes to said first address can only be performed via said first direct interface.

22. (Previously Presented) The storage system of claim 21 wherein said first memory comprises flash memory, said first direct interface is used to program said first memory, and said cache retrieves data from said first memory interface as a burst.

23. (Previously Presented) The storage system of claim 20 wherein said second memory interface also communicates with the processor via a second direct interface, which bypasses the cache for selected accesses to said second memory.

24. (Previously Presented) The storage system of claim 23 wherein when said first address is located within said second memory, writes to said first address can only be performed via said second direct interface.

25. (Previously Presented) The storage system of claim 18 wherein said cache is comprised of a plurality of lines, and when said requested data is not present in said cache, said cache fills one line with data including said requested data and fills at least one more line with data adjacent to said requested data.

26. (Previously Presented) A storage system comprising:  
a first memory interface;  
a second memory interface;  
a first memory that communicates exclusively with said first memory interface;

a second memory that communicates exclusively with said second memory interface;

an arbitration module that receives data requests from first and second processors, and processes said data requests into ordered data requests;

a cache that stores data from said first and second memories and receives one of said ordered data requests specifying a first address and associated with one processor of the first and second processors, wherein when requested data corresponding to said first address is present in said cache, said cache provides said requested data to the one processor; and

a switch that communicates with said cache, wherein when said requested data is not present in said cache, said switch exclusively connects one of said first memory interface and said second memory interface to said cache, as selected by said first address, whereby said cache retrieves said requested data,

wherein after a first time that said requested data is not present in said cache, n pages of said cache are loaded with data from sequential locations of one of said first and second memory devices to prevent any additional cache misses for as long as sequential memory locations of said one of said first and second memory devices are addressed.

27. (Previously Presented) The storage system of claim 26 wherein said cache translates an address contained within said ordered data request to obtain said first address.

28. (Previously Presented) The storage system of claim 26 wherein said first memory interface also communicates with the second processor via a direct interface, which bypasses the cache for selected accesses to said first memory.

29. (Previously Presented) The storage system of claim 28 wherein when said first address is located within said first memory, writes to said first address can only be performed via said direct interface.

30. (Previously Presented) The storage system of claim 29 wherein said first memory comprises flash memory, said first direct interface is used to program said first memory, and said cache retrieves data from said first memory interface as a burst.

31. (Previously Presented) The storage system of claim 28 wherein said second memory interface also communicates with the first and second processors via a direct arbitration interface, which bypasses the cache for selected accesses to said second memory.

32. (Previously Presented) The storage system of claim 31 wherein when said first address is located within said second memory, writes to said first address can only be performed via said second direct interface.

33. (Previously Presented) The storage system of claim 26 wherein said cache is comprised of a plurality of lines, and when said requested data is not present in said cache, said cache fills one line with data including said requested data and fills at least one more line with data adjacent to said requested data.

**X. APPENDIX B**

**EVIDENCE APPENDED**

A copy of the Office Action mailed May 7, 2007 is attached.

**XI. APPENDIX C**

**RELATED PROCEEDINGS APPENDED**

The undersigned, the Assignee, and the Appellants do not know of any other appeals or interferences which would directly affect or that would be directly affected by, or have a bearing on, the Board's decision in this Appeal.



UNITED STATES PATENT AND TRADEMARK OFFICE

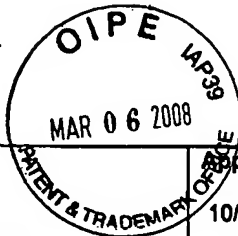
UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/646,289	08/21/2003	Son Ho	MP0390.1	9390
26703 7590 05/07/2007 HARNESSE, DICKY & PIERCE P.L.C. 5445 CORPORATE DRIVE SUITE 200 TROY, MI 48098			EXAMINER PATEL, KAUSHIKKUMAR M	
			ART UNIT 2188	PAPER NUMBER
			MAIL DATE 05/07/2007	DELIVERY MODE PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

5054-000090/CRA  
Final OA



MOW

DUE: 8-7-07

### Office Action Summary

Application No.

10/646,289

Applicant(s)

HO ET AL.

Examiner

Kaushikkumar Patel

Art Unit

2188

— The MAILING DATE of this communication appears on the cover sheet with the correspondence address —  
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

### Status

- 1) ☒ Responsive to communication(s) filed on 05 March 2007.
- 2a) ☒ This action is FINAL. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

### Disposition of Claims

- 4) ☒ Claim(s) 1-2, 4-7, 9-10, 12-15 and 17-33 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1, 2, 4-7, 9, 10, 12-15 and 17-33 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 21 August 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

### Attachment(s)

- |  |   |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892)   | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)   | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)             |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)<br>Paper No(s)/Mail Date <u>1/4/07, 1/11/07</u> . | 6) <input type="checkbox"/> Other: _____  |

**DETAILED ACTION**

***Response to Amendment***

1. This office action is in response to applicant's communication filed March 05, 2007 in response to PTO office action mailed December 05, 2006. The applicant's remarks and amendments to the claims and/or specification were considered with the results that follow.
2. In response to the last office action, claims 1, 5, 10, 13, 18 and 26 have been amended. No claims have been added. No claims have been canceled. As a result, claims 1-2, 4-7, 9-10, 12-15 and 17-33 remain pending in this application.
3. Double patenting rejections of claims had been withdrawn due to Terminal Disclaimer filed on March 05, 2007. The claim objections are also withdrawn due to applicant's remarks (page 15) filed on March 05, 2007.

***Response to Arguments***

4. Applicant's with respect to claims 1, 5, 13, 18 and 26 have been considered but are moot in view of the new ground(s) of rejection.

***Information Disclosure Statement***

5. The information disclosure statement (IDS) submitted on January 4 and 11, 2007 had considered by the examiner.



***Admitted Prior Art***

6. Applicant has not traversed the Examiner's assertion of Official Notice with regard to the rejection of claims 19 and 26 in the previous office action, therefore the well-known facts presented in these rejections are taken to be admitted prior art. These facts are summarized as follows: Computer/storage systems with cache using virtual index (virtual cache) or physical index (physical cache) as well as virtual memory addressing are well known in the art. The physical cache requires address translation for data access from cache and hence cache translating virtual to physical addressing is known in the art.

***Claim Rejections - 35 USC § 103***

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

8. Claims 1-2, 5-7, 10, 13-15, 18-19 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Zaidi et al. (US 6,601,126 B1) (Zaidi herein after), Jeddeloh (US 7,133,972 B2) and Loafman (US 2005/0021916 A1).

As per claims 1 and 18, Zaidi teaches memory storage system that is accessed by a first central processing unit (CPU) (fig. 1, item 100 and item 112, also see figs. 20-23), comprising:

a line cache including a plurality of pages that are accessed by the first CPU (fig. 1, item 126. Also caches are known to store pages accessed by CPU, see background art section of present application); and

a first memory device that stores data that is loaded into said line cache when miss occurs (fig.1, item 108 is connected to processor via cache 126 to CPU 112, column 22, lines 65-67, teaches processor use cache to access data from memory 108 and as per present application's background art section, data are loaded from lower latency memories to cache when miss occurs);

a second memory device (figs. 1-2 and 20-23, items 106, 108, 248, 250 and indicated as flash and SDRAM);

a line cache control system that controls data flow between said line cache, the first CPU, said first memory device and said second memory device (CPU accessing data from lower level storage devices through cache teaches cache control system), and that includes:

a first line cache interface that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request (figs. 20-22 shows cache of CPU is connected to memory devices through MAC, which inherently teaches interfaces and CPU accessing information from cache as explained above inherently requires generating address);

a first memory interface that communicates with said first memory device and a second memory interface that communicates with said second memory device (figs. 20-23, flash and SDRAM is connected to MAC);

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if match occurs and loads page of said line cache when miss occurs [caches are known to receive addresses from processors and comparing those addresses to stored addresses and providing data to processor if hit occurs and loading pages from lower higher latency memory in case of cache miss (see background of invention in present application)], (column 23, lines 31-34 and lines 41-45, taught as CPUs supply a request and an address, the address includes both the port, device or memory bank address and the requested memory location address. Referring to figs. 20-22, col. 23 lines 22-29, "switched channel memory controller" allows multiple DMA (and processors) to simultaneously communicate with multiple output channels. Also, col. 23, lines 40-45, suggests that CPU bus can be connected to an external flash memory through one channel and SDRAM through another channel. These statements clearly indicate that there are separate and selective communication interfaces between the devices). (switches providing separate/distinct/exclusive [limitation of claim 18] interfaces are known in the art, because a separate and independent interface avoids bus or memory bank conflict and hence increases the speed of the system, see Jeddelloh, US 7,133,972, fig. 3, col. 4, lines 30-64.)

Zaidi fails to teach loading n pages from sequential locations from memory.  
Loafman teaches when miss occurs in memory (cache); system preloads consecutive

pages from lower latency storage (Loafman, par. [0026]) into memory. It would have been obvious to one having ordinary skill in the art at the time of the invention to modify the memory storage system of Zaidi by prefetching some extra sequential pages when page fault (cache miss) occurs and the requested page is being loaded from higher latency storage to memory (cache), because during sequential access of data it is highly likely that nearby data will be accessed in near future, and by prefetching adjacent (extra) pages will avoid cache miss next time CPU references the sequential data and thus, improving the performance (see Loafman paragraphs [0011]-[0013]).

Zaidi and Loafman fail to teach loading (prefetching) n pages of cache line with the first cache miss (before second miss occurs) as required by the claim. Loafman however teaches that applications/programs read data either sequentially or randomly (also admitted by applicant, remarks submitted March 05, 2007, page 18) and if data being read randomly, prefetching may not work. He further teaches that, *"if data being read randomly and the executing program makes a request to read certain amount of data that resides on two sequential pages and if data is not already in RAM, two page faults will be raised in order to load two pages in the RAM"* (Loafman, par. [0013]). It is apparently clear from above statements that Loafman is not prefetching pages if data being read randomly, but since program requests two sequential pages (even though program accesses data randomly, it is not totally random and can access pages sequentially), two page faults will be raised (i.e. when program accessing data randomly, no prefetching occurs and hence two page faults required to load two pages), but as mentioned in par. 26, two page faults to two sequential pages, means the virtual

memory manager (VMM) infers that the data now being accessed sequentially (par. [0026], *"when program accesses two successive pages (i.e. pages 202 and 204) each using a page fault, the VMM 112 assumes that the program will continue to access data sequentially"*) and hence starts prefetching (admitted by applicant, remarks filed March 05, 2007, page 17, *"by observing the pattern used by a program"*) (Loafman, par. [0013], *"because the pages are sequential, the system may infer that the data being read sequentially; and hence, pre-fetch a block of sequential pages of data"*). Thus, it is apparently clear from above explanation, that programs read data either randomly and/or sequentially and if data being read randomly, then pre-fetching creates thrashing of cache (Loafman, pars. [0013], [0027], after reading two sequential pages using two page faults, system infers that now data is being read sequentially, hence starts pre-fetching, but the actual access is still random, so *"the blocks of pages may have been pre-fetched in vain"*), hence Loafman's system confirms that the data being read is read sequentially by having two page faults to two sequential pages before pre-fetching additional data. Thus, it is apparently clear that Loafman uses two page faults (two cache misses) to two sequential pages to confirm that data is being read sequentially. Thus, if the program reads data sequentially, then it would have been obvious to one having ordinary skill in the art at the time of the invention to start prefetching data after initial (one) miss without waiting for confirmation of whether the data being read sequentially, i.e. before a second miss occurs (as in the system of Loafman). The advantage would be to reduce number of page faults, since it is known that data is being read sequentially, there is no need for confirmation and as taught by Loafman,

prefetching works splendidly well in sequential data access and reduces page faults (Loafman, pars. [0010], [0012]), thus reducing the data access latency. Loafman teaches that as long as CPU accesses data in sequential manner, than controller keeps prefetching additional pages (2,4,8 etc. Loafman, fig. 2, items 210, 220, 230) and the next sequential page is already read ahead (pre-fetched) into the cache, hence no additional cache miss occurs.

As per claim 2, Loafman teaches that if program continue sequentially accessing prefetched pages, than prefetching more pages (four and eight and so on) into cache. (Loafman, fig. 2, paragraph [0026]).

Claims 5 and 6 are rejected under same rationales as applied to claims 1 and 2. As Loafman teaches that as long as CPU accesses data in sequential manner, than controller keeps prefetching additional pages (2,4,8 etc. Loafman, fig. 2, items 210, 220, 230) and thus next sequential address is already read ahead in the cache and hence no cache miss occurs.

As per claim 7, Loafman teaches loading 2 pages, then 4 and then 8, sequentially (Loafman, fig. 2, paragraph [0026]), which inherently teaches  $m^{\text{th}}$  page from  $n$  pages (accessing  $1^{\text{st}}$  then  $2^{\text{nd}}$  and loading 4, and then 8 pages, in case of two pages  $n = 2$  and reading  $2^{\text{nd}}$  page teaches reading  $m^{\text{th}}$  ( $2^{\text{nd}}$ ) page of two pages or  $3^{\text{rd}}$  in case of 4 pages preloaded and hence  $m$  is greater than one and less than or equal to  $n$  and prefetching 4 or 8 pages teaches additional  $n$  pages).

Claims 10 and 13-15 are rejected under same rationales as applied to claims 1-2 and 5-7 above.

As per claim 19, combination of Zaidi, Jeddelloh and Loafman teach limitations of independent claim 18, but fail to teach cache translating an address. Computer/storage systems with cache using virtual index (virtual cache) or physical index (physical cache) as well as virtual memory addressing are well known in the art. The physical caches requires address translation for data access from cache and hence cache translating virtual to physical addressing is known in the art, and examiner takes official notice of that. Physical caches are simpler to build; hence it would have been obvious to one having ordinary skill in the art at the time of the invention to use physical cache performing address translation in the system of Zaidi, Jeddelloh and Loafman.

As per claim 25, Loafman teaches loading multiple lines of data from secondary storage device to cache as explained with respect to claim 1 above.

9. Claims 4, 9, 12, 17, 26 and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Zaidi, Jeddelloh and Loafman as applied to claims 1-2, 5-7 and 13-15 above, and further in view of Barroso et al. (US 6,725,334 B2).

As per claims 4, 9, 12 and 17, Zaidi, Jeddelloh and Loafman teach a dual processor system with two caches (fig. 2, items 202 and 214, first and second processors, and items 208 and 224, two caches). Zaidi, Jeddelloh and Loafman inherently teach cache interface with first and second CPUs and both generates read requests and hence first and second address and providing an exclusive interface as taught in claim 1. Zaidi teaches system with two caches for two processors but fails to teach cache arbitration device which communicates with first and second cache

interfaces and resolves cache access conflicts between first and second CPUs. Barroso teaches a second level shared cache with switch (fig. 1, item 130 and 120), which provides interfaces with first and second CPU and arbitrates between first and second CPU (column 4, lines 10-21).

It would have been obvious to one having ordinary skill in the art at the time of invention to modify the multiple cache with multiple processor system of Zaidi and used one cache with switch as taught by Barroso to reduce the cost and the waste of the cache capacity and shared cache avoids duplication of data in individual caches (column 1, lines 45-65).

Claim 26 recites storage system with two processors and ordered data requests. Combination of claims 1 and 4, as taught above teaches dual processor system with shared cache and arbitration device. Jeddeloh teaches switch with arbitration logic to determine memory access priorities and ordering requests according to priority (Jeddeloh, col. 4, line 65 – col. 5, line 2).

As per claim 27, combination of Zaidi, Jeddeloh, Loafman and Barroso teaches limitations of independent claim 26, but fail to teach cache translating an address. Computer/storage systems with cache using virtual index (virtual cache) or physical index (physical cache) as well as virtual memory addressing are well known in the art. The physical caches requires address translation for data access from cache and hence cache translating virtual to physical addressing is known in the art, and examiner takes official notice of that. Physical caches are simpler to build; hence it would have been



obvious to one having ordinary skill in the art at the time of the invention to use physical cache performing address translation in the system of Zaidi, Jeddeloh and Loafman.

As per claim 33, Loafman teaches loading multiple lines of data from secondary storage device to cache as explained with respect to claim 1 above.

10. Claims 20-24 are rejected under **35 U.S.C. 103(a)** as being unpatentable over Zaidi, Jeddeloh and Loafman as applied to claim 18 above, and further in view of Alexander et al. (US 6,131,155).

As per claims 20 and 23, Zaidi, Jeddeloh and Loafman teach all the limitations of independent claim 18, but fail to teach direct interface to bypass cache. Alexander teaches a CPU programmed to bypass a cache when necessary (Alexander, abstract).

It would have been obvious to one having ordinary skill in the art at the time of the invention to utilize the direct access interface to memory, bypassing the cache as taught by Alexander in the system of Zaidi, Jeddeloh and Loafman, because data caches provides performance improvement only if program execution performs repeated accesses of data over a short period of time to a small group of data and large amounts of data transfers degrades the performance by the phenomenon known as thrashing, so bypassing a cache and directly reading burst data from memory increases the performance (Alexander, abstract, col. 2, lines 21-56). Also providing a direct/exclusive and independent interface avoids bus or memory bank conflict as explained with respect to claims 1, 4, 18 and 26 above.

As per limitation of claims 21-22 and 24, Alexander teaches that when necessary in order to fetch or store (reading from and writing/programming to memory device) data items directly from/to the memory, ensuring data accesses exhibiting a high degree of locality are made to the cache, while those accesses that are non-local are made directly to the main memory, bypassing the cache (Alexander, abstract), proves that depending upon addresses certain access are only performed through the direct memory interface, bypassing the cache.

11. Claims 28-32 are rejected under **35 U.S.C. 103(a)** as being unpatentable over Zaidi, Jeddeloh, Loafman and Barroso as applied to claim 26 above, and further in view of Alexander et al. (US 6,131,155).

As per claim 28, Zaidi, Jeddeloh, Loafman and Barroso teach all the limitations of independent claim 26, but fail to teach direct interface to bypass cache. Alexander teaches a CPU programmed to bypass a cache when necessary (Alexander, abstract).

It would have been obvious to one having ordinary skill in the art at the time of the invention to utilize the direct access interface to memory, bypassing the cache as taught by Alexander in the system of Zaidi, Jeddeloh, Loafman and Barroso, because data caches provides performance improvement only if program execution performs repeated accesses of data over a short period of time to a small group of data and large amounts of data transfers degrades the performance by the phenomenon known as threshing, so bypassing a cache and directly reading burst data from memory increases the performance (Alexander, abstract, col. 2, lines 21-56). Also providing a

direct/exclusive and independent interface avoids bus or memory bank conflict as explained with respect to claims 1, 4, 18 and 26 above.

As per limitation of claims 29 and 30-32, Alexander teaches that when necessary in order to fetch or store (reading from and writing/programming to memory device) data items directly from/to the memory, ensuring data accesses exhibiting a high degree of locality are made to the cache, while those accesses that are non-local are made directly to the main memory, bypassing the cache (Alexander, abstract), proves that depending upon addresses certain access are only performed through the direct memory interface, bypassing the cache.

As per limitation of claim 31, Zaidi and Barroso teach an arbiter and MAC (Zaidi, fig. 2, items 242, 244) and switch (Barroso, fig.1, item 120) to resolve memory access conflict (Zaidi, column 23, lines 40-45) but fail to teach arbiter for direct read/write interface. It would have been obvious to one having ordinary skill in the art at the time of the invention would provide arbiter for direct (bypassing cache interface) interface, because when multiple CPUs accessing memory device, providing arbitration avoids the conflict for same data.

### ***Conclusion***

12. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kaushikkumar Patel whose telephone number is 571-272-5536. The examiner can normally be reached on 8.00 am - 4.30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hyung Sough can be reached on 571-272-6799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

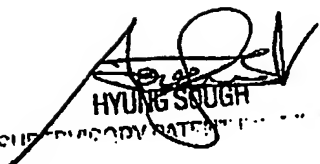
Kaushikkumar Patel

Application/Control Number: 10/646,289  
Art Unit: 2188

Page 15

  
kmp

Examiner  
Art Unit 2188

  
HYUNG SOUGH  
4-29-07